

EXHIBIT B

Alpha Architecture Reference Manual



Contributing Authors

Richard Witek

Alpha co-architect

and

Ellen M. Batbouta

Richard A. Brunner

Wayne M. Cardoza

Daniel W. Dobberpuhl

Robert A. Giggi

Henry N. Grieb

Richard B. Grove

Robert H. Halstead, Jr.

Michael S. Harvey

Nancy P. Kronenberg

Raymond J. Lanza

Stephen J. Morris

William B. Noyce

Charles G. Nylander

Mary H. Payne

Audrey R. Reith

Robert M. Supnik

Benjamin J. Thomas

Catharine Van Ingen

Edited by

Richard L. Sites

Alpha co-architect

digital

DIGITAL PRESS

Copyright © 1992 by Digital Equipment Corporation

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without prior written permission of the publisher.

Printed in the United States of America.

9 8 7 6 5 4 3 2 1

Order number EY-L520E-DP

ISBN 1-55558-098-X

Technical Writer: Charles Greenman

Production Editor: Kathe Rhoades

Technical Illustrator: Lynne Kenison

Cover Design: Marshall Henrichs

The following are trademarks of Digital Equipment Corporation: DEC, the Digital logo, OpenVMS, PALcode, PDP-11, VAX, VMS, and ULTRIX. Cray is a registered trademark of Cray Research, Inc. IBM is a registered trademark of International Business Machines Corporation. OSF/1 is a registered trademark of Open Software Foundation, Inc. UNIX is a registered trademark of UNIX System Laboratories, Inc.

Digital believes the information in this book is accurate as of its publication date; such information is subject to change without notice. Digital is not responsible for any inadvertent errors.

4.3 Control Instructions

Alpha provides integer conditional branch, unconditional branch, branch to subroutine, and jump instructions. The PC used in these instructions is the updated PC, as described in Section 3.1.1.

To allow implementations to achieve high performance, the Alpha architecture includes explicit hints based on a branch-prediction model:

1. For many implementations of computed branches (JSR/RET/JMP), there is a substantial performance gain in forming a good guess of the expected target I-cache address before register Rb is accessed.
2. For many implementations, the first-level (or only) I-cache is no bigger than a page (8 KB to 64 KB).
3. Correctly predicting subroutine returns is important for good performance. Some implementations will therefore keep a small stack of predicted subroutine return I-cache addresses.

The Alpha architecture provides three kinds of branch-prediction hints: likely target address, return-address stack action, and conditional branch-taken.

For computed branches, the otherwise unused displacement field contains a function code (JMP/JSR/RET/JSR_COROUTINE), and, for JSR and JMP, a field that statically specifies the 16 low bits of the most likely target address. The PC-relative calculation using these bits can be exactly the PC-relative calculation used in unconditional branches. The low 16 bits are enough to specify an I-cache block within the largest possible Alpha page and hence are expected to be enough for branch-prediction logic to start an early I-cache access for the most likely target.

For all branches, hint or opcode bits are used to distinguish simple branches, subroutine calls, subroutine returns, and coroutine links. These distinctions allow branch-predict logic to maintain an accurate stack of predicted return addresses.

For conditional branches, the sign of the target displacement is used as a taken/fall-through hint. The instructions are summarized in Table 4-3.

Table 4–3: Control Instructions Summary

Mnemonic	Operation
BEQ	Branch if Register Equal to Zero
BGE	Branch if Register Greater Than or Equal to Zero
BGT	Branch if Register Greater Than Zero
BLBC	Branch if Register Low Bit Is Clear
BLBS	Branch if Register Low Bit Is Set
BLE	Branch if Register Less Than or Equal to Zero
BLT	Branch if Register Less Than Zero
BNE	Branch if Register Not Equal to Zero
BR	Unconditional Branch
BSR	Branch to Subroutine
JMP	Jump
JSR	Jump to Subroutine
RET	Return from Subroutine
JSR_COROUTINE	Jump to Subroutine Return

4.3.1 Conditional Branch

Format:

Bxx Ra.rq,disp.al !Branch format

Operation:

```
{update PC}
va ← PC + {4*SEXT(disp)}
IF TEST(Rav, Condition_based_on_Opcode) THEN
    PC ← va
```

Exceptions:

None

Instruction mnemonics:

BEQ	Branch if Register Equal to Zero
BGE	Branch if Register Greater Than or Equal to Zero
BGT	Branch if Register Greater Than Zero
BLBC	Branch if Register Low Bit Is Clear
BLBS	Branch if Register Low Bit Is Set
BLE	Branch if Register Less Than or Equal to Zero
BLT	Branch if Register Less Than Zero
BNE	Branch if Register Not Equal to Zero

Qualifiers:

None

Description:

Register Ra is tested. If the specified relationship is true, the PC is loaded with the target virtual address; otherwise, execution continues with the next sequential instruction.

The displacement is treated as a signed longword offset. This means it is shifted left two bits (to address a longword boundary), sign-extended to 64 bits, and added to the updated PC to form the target virtual address.

The conditional branch instructions are PC-relative only. The 21-bit signed displacement gives a forward/backward branch distance of +/- 1M instructions.

The test is on the signed quadword integer interpretation of the register contents; all 64 bits are tested.

Notes:

- Forward conditional branches (positive displacement) are predicted to fall through. Backward conditional branches (negative displacement) are predicted to be taken. Conditional branches do not affect a predicted return address stack.

4.4.1 Longword Add

Format:

ADDL	Ra.rq,Rb.rq,Rc.wq	!Operate format
ADDL	Ra.rq,#b.ib,Rc.wq	!Operate format

Operation:

$Rc \leftarrow \text{SEXT}((Rav + Rbv) < 31:0 >)$

Exceptions:

Integer Overflow

Instruction mnemonics:

ADDL Add Longword

Qualifiers:

Integer Overflow Enable (/V)

Description:

Register Ra is added to register Rb or a literal, and the sign-extended 32-bit sum is written to Rc.

The high order 32 bits of Ra and Rb are ignored. Rc is a proper sign extension of the truncated 32-bit sum. Overflow detection is based on the longword sum $Rav < 31:0 > + Rbv < 31:0 >$.

4.4.5 Integer Signed Compare

Format:

CMPxx	Ra.rq,Rb.rq,Rc.wq	!Operate format
CMPxx	Ra.rq,#b.ib,Rc.wq	!Operate format

Operation:

```
IF  Rav SIGNED_RELATION Rbv THEN
    Rc ← 1
ELSE
    Rc ← 0
```

Exceptions:

None

Instruction mnemonics:

CMPEQ	Compare Signed Quadword Equal
CMPLE	Compare Signed Quadword Less Than or Equal
CMPLT	Compare Signed Quadword Less Than

Qualifiers:

None

Description:

Register Ra is compared to Register Rb or a literal. If the specified relationship is true, the value one is written to register Rc; otherwise, zero is written to Rc.

Notes:

- Compare Less Than A,B is the same as Compare Greater Than B,A; Compare Less Than or Equal A,B is the same as Compare Greater Than or Equal B,A. Therefore, only the less-than operations are included.